A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks

Hadi Salman∗
Microsoft Research AI
hadi.salman@microsoft.com

Greg Yang
Microsoft Research AI
gregyang@microsoft.com

Huan Zhang
UCLA
huan@huan-zhang.com

Cho-Jui Hsieh
UCLA
chohsieh@cs.ucla.edu

Pengchuan Zhang
Microsoft Research AI
penzhan@microsoft.com

Abstract

Verification of neural networks enables us to gauge their robustness against adversarial attacks. Verification algorithms fall into two categories: exact verifiers that run in exponential time and relaxed verifiers that are efficient but incomplete. In this paper, we unify all existing LP-relaxed verifiers, to the best of our knowledge, under a general convex relaxation framework. This framework works for neural networks with diverse architectures and nonlinearities and covers both primal and dual views of neural network verification. Next, we perform large-scale experiments, amounting to more than 22 CPU-years, to obtain exact solution to the convex-relaxed problem that is optimal within our framework for ReLU networks. We find the exact solution does not significantly improve upon the gap between PGD and existing relaxed verifiers for various networks trained normally or robustly on MNIST and CIFAR datasets. Our results suggest there is an inherent barrier to tight verification for the large class of methods captured by our framework. We discuss possible causes of this barrier and potential future directions for bypassing it. Our code and trained models are available at http://github.com/Hadisalman/robust-verify-benchmark.

1 Introduction

A classification neural network \( f : \mathbb{R}^n \rightarrow \mathbb{R}^K \) (where \( f_i(x) \) should be thought of as the \( i \)th logit) is considered adversarially robust with respect to an input \( x \) and its neighborhood \( S_{\text{in}}(x) \) if

\[
\min_{x' \in S_{\text{in}}(x), i \neq i^*} f_i(x) - f_{i^*}(x') > 0, \quad \text{where} \quad i^* = \arg \max_j f_j(x).
\]

Many recent works have proposed robustness verification methods by lower-bounding eq. (1); the positivity of this lower bound proves the robustness w.r.t. \( S_{\text{in}}(x) \). A dominant approach thus far has tried to relax eq. (1) into a convex optimization problem, from either the primal view [Zhang et al., 2018, Gehr et al., 2018, Singh et al., 2018, Weng et al., 2018] or the dual view [Wong and Kolter, 2018, Dvijotham et al., 2018b, Wang et al., 2018b]. In our first main contribution, we propose a layer-wise convex relaxation framework that unifies these works and reveals the relationships between them (Fig. 1). We further show that the performance of methods within this framework is subject to a theoretical limit: the performance of the optimal layer-wise convex relaxation.

This then begs the question: is the road to fast and accurate robustness verification paved by just faster and more accurate layer-wise convex relaxation that approaches the theoretical limit? In our second main contribution, we answer this question in the negative. We perform extensive experiments

∗Work done as part of the Microsoft AI Residency Program.

Please see http://arxiv.org/abs/1902.08722 for the full and most recent version of this paper.

Figure 1: Relationship between existing relaxed algorithms and our framework. See Appendix D for detailed discussions of each unlabeled arrow from the “Primal view” side.

with deep ReLU networks to compute the optimal layer-wise convex relaxation and compare with the LP-relaxed dual formulation from Wong and Kolter [2018], the PGD attack from Madry et al. [2017], and the mixed integer linear programming (MILP) exact verifier from Tjeng et al. [2019].

Over different models, sizes, training methods, and datasets (MNIST and CIFAR-10), we find that (i) in terms of lower bounding the minimum $\ell_{\infty}$ adversarial distortion\(^3\), the optimal layer-wise convex relaxation only slightly improves the lower bound found by Wong and Kolter [2018], especially when compared with the upper bound provided by the PGD attack, which is consistently 1.5 to 5 times larger; (ii) in terms of upper bounding the robust error, the optimal layer-wise convex relaxation does not significantly close the gap between the PGD lower bound (or MILP exact answer) and the upper bound from Wong and Kolter [2018]. Therefore, there seems to be an inherent barrier blocking our progress on this road of layer-wise convex relaxation, and we hope this work provokes much thought in the community on how to bypass it.

2 Preliminaries and Related Work

Exact verifiers and NP-completeness. For ReLU networks (piece-wise linear networks in general), exact verifiers solve the robustness verification problem (1) by typically employing MILP solvers [Cheng et al., 2017, Lomuscio and Maganti, 2017, Dutta et al., 2018, Fischetti and Jo, 2017, Tjeng et al., 2019, Xiao et al., 2019] or Satisfiability Modulo Theories (SMT) solvers [Scheibler et al., 2015, Katz et al., 2017, Carlini et al., 2017, Ehlers, 2017]. However, due to the NP-completeness for solving such a problem [Katz et al., 2017, Weng et al., 2018], it can be really challenging to scale these to large networks. It can take Reluplex [Katz et al., 2017] several hours to find the minimum distortion of an example for a ReLU network with 5 inputs, 5 outputs, and 300 neurons. A recent work by Tjeng et al. [2019] uses MILP to exactly verify medium-size networks, but the verification time is very sensitive to how a network is trained; for example, it is fast for networks trained using the LP-relaxed dual formulation of Wong and Kolter [2018], but much slower for normally trained networks. A concurrent work by Xiao et al. [2019] trains networks with the objective of speeding up the MILP verification problem, but this compromises on the performance of the network.

Relaxed and efficient verifiers. These verifiers solve a relaxed, but more computationally efficient, version of (1), and have been proposed from different perspectives. From the primal view, one can relax the nonlinearity in (1) into linear inequality constraints. This perspective has been previously explored as in the framework of “abstract transformers” [Singh et al., 2018, 2019a,b, Gehr et al.,

\(^3\)The radius of the largest $\ell_{\infty}$ ball in which no adversarial examples can be found.
While our framework does not cover this SDP relaxation, it is not clear to us how to extend the optimization domain where the optimization domain is the set of activations and preactivations in the linear layer, and is the constraint on in the adversarial attack model. We also assume that is an -layer feedforward NN. For notational simplicity, we denote by and by . We define as
\[ x^{(l+1)} = \sigma^{(l)}(W^{(l)}x^{(l)} + b^{(l)}) \quad \forall l \in [L], \quad \text{and} \quad f(x) := z^{(L)} = W^{(L)}x^{(L)} + b^{(L)}, \]
where is the input, is the weight matrix and bias vector of the -th linear layer, and is a (nonlinear) activation function like leaky-ReLU, the sigmoid family (including sigmoid, arctan, hyperbolic tangent, etc), and the pooling family (MaxPool, AvgPool, etc). Our results can be easily extended to networks with convolutional layers and skip connections as well, similar to what is done in [Wong et al., 2018], as these can be seen as special forms of (2).

Consider the following optimization problem \( O(c, c_0, L, z^{[L]}, \overline{z}^{[L]}) : \)
\[
\min_{(x^{[L+1]}, x^{[L]}) \in \mathcal{D}} c^T x^{[L]} + c_0
\]
\[
\text{s.t.} \quad z^{(l)} = W^{(l)}x^{(l)} + b^{(l)}, l \in [L],
\]
\[
x^{(l+1)} = \sigma^{(l)}(z^{(l)}), l \in [L],
\]
where the optimization domain is the set of activations and preactivations \( \{x^{(0)}, x^{(1)}, \ldots, x^{(L)}, z^{(0)}, z^{(1)}, \ldots, z^{(L-1)}\} \) satisfying the bounds \( z^{(l)} \leq z^{(l)} \leq \overline{z}^{(l)} \forall l \in [L] \), i.e.,
\[
\mathcal{D} = \{ (x^{[L+1]}, z^{[L]}) : x^{(0)} \in S_{in}(x^{nom}), \quad z^{(l)} \leq z^{(l)} \leq \overline{z}^{(l)} \forall l \in [L] \}. \quad (3)
\]
If \( c^T = W^{(L)}_{i,i} - W^{(L)}_{i,i}, c_0 = b^{(L)}_{i} - b^{(L)}_{i}, z^{[L]} = -\infty, \) and \( \overline{z}^{[L]} = \infty \), then \( (O) \) is equivalent to problem (1). However, when we have better information about valid bounds \( z^{(l)} \) and \( \overline{z}^{(l)} \) of \( z^{(l)} \), we can significantly narrow down the optimization domain and, as will be detailed shortly, achieve tighter solutions when we relax the nonlinearities. We denote the minimal value of \( O(c, c_0, L, z^{[L]}, \overline{z}^{[L]}) \) by \( p^*(c, c_0, L, z^{[L]}, \overline{z}^{[L]}) \), or just \( p^*_O \) when no confusion arises.

**Obtaining lower and upper bounds \( (z^{[L]}, \overline{z}^{[L]}) \) by solving sub-problems.** This can be done by recursively solving \( (O) \) with specific choices of \( c \) and \( c_0 \), which is a common technique used in many works [Wong and Kolter, 2018, Dvijotham et al., 2018b]. For example, one can obtain a lower bound of \( z^{(l)} \), by solving \( O(W^{(l)}_{i,i}, b^{(l)}_{i}, \ell, z^{(l)}, \overline{z}^{(l)}) \); this shows that one can estimate \( z^{(l)} \) and \( \overline{z}^{(l)} \).
We emphasize that by optimal, we mean the optimal convex relaxation of the single nonlinear constraint $x^{(l+1)} = \sigma^{(l)}(z^{(l)})$ (see Proposition (B.3)) instead of the optimal convex relaxation of the nonconvex feasible set of the original problem $\mathcal{O}$. As such, techniques as in [Anderson et al., 2018, Raghunathan et al., 2018b] are outside our framework; see appendix C for more discussions.

**Greedily solving the primal with linear bounds.** As another special case, when there are exactly one linear upper bound and one linear lower bound for each nonlinear layer in (C) as follows:

$$
\tilde{\sigma}^{(l)}(z^{(l)}) := \tilde{\sigma}^{(l)} z^{(l)} + \tilde{b}^{(l)}, \quad \bar{\sigma}^{(l)}(z^{(l)}) := \bar{\sigma}^{(l)} z^{(l)} + \bar{b}^{(l)}.
$$

the objective $p^*_C$ can be greedily bounded in a layer-by-layer manner. We can derive one linear upper and one linear lower bound of $z^L := c^T x^L + c_0$ with respect to $z^{(L-1)}$, using the fact that $z^L = \tilde{c}^T \tilde{\sigma}^{(L-1)}(z^{(L-1)}) + c_0$ and that $\bar{\sigma}^{(L-1)}(z^{(L-1)})$ is linearly upper and lower bounded by
Theorem 4.2. Assume that the nonlinear layer $\sigma^{(l)}$ is non-interactive (definition B.2) and the optimal layer-wise relaxation $\underline{\sigma}^{(l)}_{opt}$ and $\overline{\sigma}^{(l)}_{opt}$ are defined in (6). Then the lower bound $d^*_C$ provided by the dual of the optimal layer-wise convex-relaxed problem (9) and $d^*_O$ provided by the dual of the original problem (11) are the same.
The complete proof is in Appendix F. Theorem 4.2 combined with the strong duality result of Theorem 4.1 implies that the primal relaxation (C) and the two kinds of dual relaxations, (9) and (11), are all blocked by the same barrier. As concrete examples:

**Corollary 4.3** \( p_{\text{C opt}}^* = d_{\text{C opt}}^* \). Suppose that the nonlinear activation functions \( \sigma(l) \) for all \( l \in [L] \) are (for example) among the following: ReLU, step, ELU, sigmoid, tanh, polynomials and max pooling with disjoint windows. Assume that \( \sigma_{\text{opt}}(l) \) and \( \sigma_{\text{opt}}(l) \) are defined in (6), respectively. Then we have that the lower bound \( p_{\text{C opt}}^* \) provided by the primal optimal layer-wise relaxation (C) and \( d_{\text{C opt}}^* \) provided by the dual relaxation (11) are the same.

**Greedily solving the dual with linear bounds.** When the relaxed bounds \( p \) and \( \sigma \) are linear as defined in (7), the dual objective (9) can be lower bounded as below:

\[
p_{\text{C opt}}^* = d_{\text{C opt}}^* \geq \sum_{l=0}^{L-1} \left( b_l^T \lambda(l) + -b_l^T \lambda(l) - b_l^T \mu(l) \right) + c_0 - \sup_{x \in S_{\lambda(l)}(x^{\text{nom}})} \left( W_l^T \mu(l) \right)^T x,
\]

where the dual variables \( \lbrace \mu(l), \lambda(l) \rbrace \) are determined by a backward propagation

\[
\lambda(l-1) = -c, \quad \mu(l) = a(l) (\lambda(l))_+ + \sigma(l) (\lambda(l))_-, \quad \lambda(l-1) = W_l^T \mu(l) \quad \forall l \in [L-1],
\]

We provide the derivation of this algorithm in Appendix G. It turns out that this algorithm can exactly recover the algorithm proposed in Wong and Kolter [2018], where

\[
\alpha(l)(z(l)) := \alpha(l) z(l), \quad \sigma(l)(z(l)) := \sigma(l) z(l) - z(l),
\]

and \( 0 \leq \alpha(l) \leq 1 \) represents the slope of the lower bound. When \( \alpha(l) = \frac{\sigma(l)}{\sigma(l) - z(l)} \), the greedy algorithm also recovers Fast-Lin [Weng et al., 2018], which explains the arrow from Wong and Kolter [2018] to Weng et al. [2018] in Fig. 1. When \( \alpha(l) \) is chosen adaptively as in CROWN [Zhang et al., 2018], the greedy algorithm then recovers CROWN, which explains the arrow from Wong and Kolter [2018] to Zhang et al. [2018] in Fig. 1. See Appendix D for more discussions on the relationship between the primal and dual greedy solvers.

## 5 Optimal LP-relaxed Verification

In the previous sections, we presented a framework that subsumes all existing layer-wise convex-relaxed verification algorithms except that of Raghunathan et al. [2018b]. For ReLU networks, being piece-wise linear, these correspond exactly to the set of all existing LP-relaxed algorithms, as discussed above. We showed the existence of a barrier, \( p_{\text{C opt}}^* \), that limits all such algorithms. Is this just theoretical babbling or is this barrier actually problematic in practice?

In the next section, we perform extensive experiments on deep ReLU networks, evaluating the tightest convex relaxation afforded by our framework (denoted LP-ALL) against a greedy dual algorithm (Algorithm 1 of Wong and Kolter [2018], denoted LP-GREEDY) as well as another algorithm LP-LAST, intermediate in speed and accuracy between them. Both LP-GREEDY and LP-LAST solve the bounds \( z[l], z[l] \) by setting the dual variables heuristically (see previous section), but LP-GREEDY solves the adversarial loss in the same manner while LP-LAST solves this final LP exactly. We also compare them with the opposite bounds provided by PGD attack [Madry et al., 2017], as well as exact results from MILP [Tjeng et al., 2019].

For the rest of the main text, we are only concerned with ReLU networks, so (C) subject to (4) is in fact an LP.

---

Footnotes:
1. Theorem 2 in Dvijotham et al. [2018b] is a special case of our Theorem 4.2, when applied to ReLU networks. Our proof makes use of the Fenchel-Moreau theorem to deal with general nonlinearities, which is different from that in Dvijotham et al. [2018b].
2. Note that in practice (as in [Tjeng et al., 2019]), MILP has a time budget, and usually not every sample can be verified within that budget, so that in the end we still obtain only lower and upper bounds given by samples verified to be robust or nonrobust.
5.1 LP-ALL Implementation Details

In order to exactly solve the tightest LP-relaxed verification problem of a ReLU network, two steps are required: (A) obtaining the tightest pre-activation upper and lower bounds of all the neurons in the NN, excluding those in the last layer, then (B) solving the LP-relaxed verification problem exactly for the last layer of the NN.

Step A: Obtaining Pre-activation Bounds. This can be done by solving sub-problems of the original relaxed problem \((C)\) subject to (4). Given a NN with \(L_0\) layers, for each layer \(l_0 \in [L_0]\), we obtain a lower (resp. upper) bound \(z_j^{(l_0)}\) (resp. \(\bar{z}_j^{(l_0)}\)) of \(z_j^{(l_0)}\), for all neurons \(j \in [n^{(l_0)}]\). We do this by setting

\[
L \leftarrow l_0, \quad c^\top \leftarrow W_{j;0}^{(l_0)} \quad \text{(resp.} \quad c^\top \leftarrow -W_{j;0}^{(l_0)}, \quad b_j^{(l_0)} \leftarrow b_j^{(l_0)} \quad \text{(resp.} \quad c_0 \leftarrow -b_j^{(l_0)})
\]

in \((C)\) and computing the exact optimum. However, we need to solve an LP for each neuron, and practical networks can have millions of them. We utilize the fact that in each layer \(l_0\), computing the bounds \(z_j^{(l_0)}\) and \(\bar{z}_j^{(l_0)}\) for each \(j \in [n^{(l_0)}]\) can proceed independently in parallel. Indeed, we design a scheduler to do so on a cluster with 1000 CPU-nodes. See Appendix J for details.

Step B: Solving the LP-relaxed Problem for the Last Layer. After obtaining the pre-activation bounds on all neurons in the network using step (A), we solve the LP in \((C)\) subject to (4) for all \(j \in [n^{(L_0)}]\setminus \{j_{\text{nom}}\}\) obtained by setting

\[
L \leftarrow L_0, \quad c^\top \leftarrow W_{j;0}^{(L_0)} - W_{j;0}^{(L_0)}, \quad c_0 \leftarrow b_j^{(L_0)} - b_j^{(L_0)}
\]

again in \((C)\) and computing the exact minimum. Here, \(j_{\text{nom}}\) is the true label of the data point \(x_{\text{nom}}\) at which we are verifying the network. We can certify the network is robust around \(x_{\text{nom}}\) iff the solutions of all such LPs are positive, i.e. we cannot make the true class logit lower than any other logits. Again, note that these LPs are also independent of each other, so we can solve them in parallel.

Given any \(x_{\text{nom}}\), LP-ALL follows steps (A) then (B) to produce a certificate whether the network is robust around a given datapoint or not. LP-LAST on the other hand solves only step (B), and instead of doing (A), it finds the preactivation bounds greedily as in Algorithm 1 of Wong and Kolter [2018].

6 Experiments

We conduct two experiments to assess the tightness of LP-ALL: 1) finding certified upper bounds on the robust error of several NN classifiers, 2) finding certified lower bounds on the minimum adversarial distortion \(\epsilon\) using different algorithms. All experiments are conducted on MNIST and/or CIFAR-10 datasets.

Architectures. We conduct experiments on a range of ReLU-activated feedforward networks. MLP-A and MLP-B refer to multilayer perceptrons: MLP-A has 1 hidden layer with 500 neurons, and MLP-B has 2 hidden layers with 100 neurons each. CNN-SMALL, CNN-WIDE-K, and CNN-DEEP-K are the ConvNet architectures used in Wong et al. [2018]. Full details are in Appendix I.1.

Training Modes. We conduct experiments on networks trained with a regular cross-entropy (CE) loss function and networks trained to be robust. These networks are identified by a prefix corresponding to the method used to train them: LPD when the LP-relaxed dual formulation of Wong and Kolter [2018] is used for robust training, ADV when adversarial examples generated using PGD are used for robust training, as in Madry et al. [2017], and NOR when the network is normally trained using the CE loss function. Training details are in Appendix I.2.

Experimental Setup. We run experiments on a cluster with 1000 CPU-nodes. The total run time amounts to more than 22 CPU-years. Appendix J provides additional details about the computational resources and the scheduling scheme used, and Appendix K provides statistics of the verification time in these experiments.
Table 1: Certified bounds on the robust error on the test set of MNIST for normally and robustly trained networks. The prefix of each network corresponds to the training method used: **ADV** for PGD training [Madry et al., 2017], **NOR** for normal CE loss training, and **LPD** when the LP-relaxed dual formulation of Wong and Kolter [2018] is used for robust training.

<table>
<thead>
<tr>
<th>Network</th>
<th>( \epsilon )</th>
<th>Test Error</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>PGD</td>
<td>MILP</td>
</tr>
<tr>
<td>ADV-MLP-B</td>
<td>0.03</td>
<td>1.53%</td>
<td>4.17%</td>
<td>4.18%</td>
</tr>
<tr>
<td>ADV-MLP-B</td>
<td>0.05</td>
<td>1.62%</td>
<td>6.06%</td>
<td>6.11%</td>
</tr>
<tr>
<td>ADV-MLP-B</td>
<td>0.1</td>
<td>3.33%</td>
<td>15.86%</td>
<td>16.25%</td>
</tr>
<tr>
<td>ADV-MLP-A</td>
<td>0.1</td>
<td>4.18%</td>
<td>11.51%</td>
<td>14.36%</td>
</tr>
<tr>
<td>NOR-MLP-B</td>
<td>0.02</td>
<td>2.05%</td>
<td>10.06%</td>
<td>10.16%</td>
</tr>
<tr>
<td>NOR-MLP-B</td>
<td>0.03</td>
<td>2.05%</td>
<td>20.37%</td>
<td>20.43%</td>
</tr>
<tr>
<td>LPD-MLP-B</td>
<td>0.05</td>
<td>2.05%</td>
<td>53.37%</td>
<td>53.37%</td>
</tr>
<tr>
<td>LPD-MLP-B</td>
<td>0.1</td>
<td>4.09%</td>
<td>13.39%</td>
<td>14.45%</td>
</tr>
<tr>
<td>LPD-MLP-B</td>
<td>0.2</td>
<td>15.72%</td>
<td>33.85%</td>
<td>36.33%</td>
</tr>
<tr>
<td>LPD-MLP-B</td>
<td>0.3</td>
<td>39.22%</td>
<td>57.29%</td>
<td>59.85%</td>
</tr>
<tr>
<td>LPD-MLP-B</td>
<td>0.4</td>
<td>67.97%</td>
<td>81.85%</td>
<td>83.17%</td>
</tr>
</tbody>
</table>

### 6.1 Certified Bounds on the Robust Error

Table 1 presents the clean test errors and (upper and lower) bounds on the true robust errors for a range of classifiers trained with different procedures on MNIST. For both ADV- and LPD-trained networks, the \( \epsilon \) in Table 1 denotes the \( l_\infty \)-norm bound used for training and robust testing; for NORmally-trained networks, \( \epsilon \) is only used for the latter.

Lower bounds on the robust error are calculated by finding adversarial examples for inputs that are not robust. This is done by using PGD, a strong first-order attack, or using MILP [Tjeng et al., 2019]. Upper bounds on the robust error are calculated by providing certificates of robustness for input that is robust. This is done using MILP, the dual formulation (LP-GREEDY) presented by Wong and Kolter [2018], or our LP-ALL algorithm.

For the MILP results, we use the code accompanying the paper by Tjeng et al. [2019]. We run the code in parallel on a cluster with 1000 CPU-nodes, and set the MILP solver’s time limit to 3600 seconds. Note that this time limit is reached for ADV and NOR, and therefore the upper and lower bounds are separated by a gap that is especially large for some of the NORmally-trained networks. On the other hand, for LPD-trained networks, the MILP solver finishes within the time limit, and thus the upper and lower bounds match.

**Results.** For all NORmally and ADV-trained networks, we see that the certified upper bounds using LP-GREEDY and LP-ALL are very loose when we compare the gap between them to the lower bounds found by PGD and MILP. As a sanity check, note that LP-ALL gives a tighter bound than LP-GREEDY in each case, as one would expect. Yet this improvement is not significant enough to close the gap with the lower bounds.

This sanity check also passes for LPD-trained networks, where the LP-GREEDY-certified robust error upper bound is, as expected, much closer to the true error (given by MILP here) than for other networks. For \( \epsilon = 0.1 \), the improvement of LP-ALL-certified upper bound over LP-GREEDY is at most modest, and the PGD lower bound is tighter to the true error. For large \( \epsilon \), the improvement is much more significant in relative terms, but the absolute improvement is only 4 – 7%. In this large \( \epsilon \) regime, however, both the clean and robust errors are quite large, so the tightness of LP-ALL is less useful.

### 6.2 Certified Bounds on the Minimum Adversarial Distortion \( \epsilon \)

We are interested in searching for the minimum adversarial distortion \( \epsilon \), which is the radius of the largest \( l_\infty \) ball in which no adversarial examples can be crafted. An upper bound on \( \epsilon \) is calculated using PGD, and lower bounds are calculated using LP-GREEDY, LP-LAST, or our LP-ALL, all via binary search. Since solving LP-ALL is expensive, we find the \( \epsilon \)-bounds only for ten samples of the MNIST and CIFAR-10 datasets. In this experiment, both ADV- and LPD-networks are trained with an \( l_\infty \) maximum allowed perturbation of 0.1 and \( 8/255 \) on MNIST and CIFAR-10, respectively. See Appendix L.1 for details. Fig. 3 and 8 in the Appendix show the median percentage gap (defined in
Figure 3: The median percentage gap between the convex-relaxed algorithms (LP-ALL, LP-LAST, and LP-GREEDY) and PGD estimates of the minimum adversarial distortion $\epsilon$ on ten samples of MNIST. The error bars correspond to 95% confidence intervals. We highlight the $1.5 \times$ and $5 \times$ gaps between the $\epsilon$ value estimated by PGD, and those estimated by the LP-relaxed algorithms. For more details, please refer to Table 2 in Appendix L.2.

Appendix L.2) between the convex-relaxed algorithms and PGD bounds of $\epsilon$ for MNIST and CIFAR, respectively. Details are reported in Tables 2 and 3 in Appendix L.2.

On MNIST, the results show that for all networks trained normally or via Adv, the certified lower bounds on $\epsilon$ are 1.5 to 5 times smaller than the upper bound found by PGD; for LPD trained networks, below 1.5 times smaller. On CIFAR-10, the bounds are between 1.5 and 2 times smaller across all models. The smaller gap for LPD is of course as expected following similar observations in prior work [Wong and Kolter, 2018, Tjeng et al., 2019]. Furthermore, the improvement of LP-ALL and LP-LAST over LP-GREEDY is not significant enough to close the gap with the PGD upper bound. Note that similar results hold as well for randomly initialized networks (no training). To avoid clutter, we report these in Appendix M.

7 Conclusions and Discussions

In this work, we first presented a layer-wise convex relaxation framework that unifies all previous LP-relaxed verifiers, in both primal and dual spaces. Then we performed extensive experiments to show that even the optimal convex relaxation for ReLU networks in this framework cannot obtain tight bounds on the robust error in all cases we consider here. Thus any method will face a convex relaxation barrier as soon as it can be described by our framework. We look at how to bypass this barrier in Appendix A.

Note that different applications have different requirements for the tightness of the verification, so our barrier could be a problem for some but not for others. In so far as the ultimate goal of robustness verification is to construct a training method to lower certified error, this barrier is not necessarily problematic — some such method could still produce networks for which convex relaxation as described by our framework produces accurate robust error bounds. An example is the recent work of Gowal et al. [2018] which shows that interval bound propagation, which often leads to loose certification bounds, can still be used for verified training, and is able to achieve state-of-the-art verified accuracy when carefully tuned. However, without a doubt, in all cases, tighter estimates should lead to better results, and we reveal a definitive ceiling on most current methods.
References


